//功能：给定平面数据点的最小二乘B样条曲线逼近。
//输入参数：(m_xDVertex, m_yDVertex)-双精度数组给出的数据点；数据点数由m_xDVertex.GetSize()确定；m_aU-数据点参数化给出的双精度数组；m_nCtrl0-控制顶点数；m_Degree-逼近曲线次数。m_aNode-节点矢量双精度数组，求B样条基函数调用。都是受保护成员或公有成员。
//输出参数：(m_xAVertex, m_yAVertex)- B样条逼近曲线的控制顶点，受保护成员；Singular-若为true,表示线性方程组系数矩阵奇异。
//调用函数：GetBaseFunVal-求B样条基函数，见7.1.2节；agjdn-全选主元高斯-约当消去法解线性代数方程组，参考文献[79]改造而来。

```
void BsplineCuvApprox()
{
	int i0,i1;
	int n= m_xDVertex.GetSize()-1;
	CArray<double,double> Rx,Ry,a,b;
	CArray<CArray<double,double>,CArray<double,double>&> N;
	CArray<CArray<double,double>,CArray<double,double>&> F;
	a.SetSize((m_nCtrl0-1)*(m_nCtrl0-1));
	b.SetSize(2*(m_nCtrl0-1));
	N.SetSize(n-1);
	for(i0=0;i0<=n-2;i0++) N[i0].SetSize(m_nCtrl0-1);
	Rx.SetSize(n-1);	Ry.SetSize(n-1);
	m_xAVertex.SetSize(m_nCtrl0+1);  m_yAVertex.SetSize(m_nCtrl0+1);
	F.SetSize(m_nCtrl0-1);
	for(i0=0;i0<=m_nCtrl0-2;i0++) F[i0].SetSize(m_nCtrl0-1);
	for(i0=1;i0<=n-1;i0++)
	{
		for(int j=1;j<=m_nCtrl0-1;j++)
		{
			N[i0-1][j-1]=GetBaseFunVal(m_aU[i0],j,m_Degree);
		}
	}
	for(i0=0;i0<=m_nCtrl0-2;i0++)
	{
		for(int j=0;j<=m_nCtrl0-2;j++)
		{
			F[i0][j]=0.;
			for(int jj=0;jj<=n-2;jj++)
			{
				F[i0][j]=F[i0][j]+N[jj][i0]*N[jj][j];
			}
		}
	}
	for(i1=0;i1<=m_nCtrl0-2;i1++)
	{
		for(int j1=0;j1<=m_nCtrl0-2;j1++)	a[i1*(m_nCtrl0-1)+j1]=F[i1][j1];
	}
	for(i0=1;i0<=n-1;i0++)
	{
		double B0=GetBaseFunVal(m_aU[i0],0,m_Degree);
		double Bn=GetBaseFunVal(m_aU[i0],m_nCtrl0,m_Degree);
		Rx[i0-1]=m_xDVertex[i0]-(B0*m_xDVertex[0]+Bn*m_xDVertex[n]);
		Ry[i0-1]=m_yDVertex[i0]-(B0*m_yDVertex[0]+Bn*m_yDVertex[n]);
	}
	for(int j=0;j<=m_nCtrl0-2;j++)
	{
		m_xAVertex[j]=0.;
		m_yAVertex[j]=0.;
		for(int i1=0;i1<=n-2;i1++)
		{
			m_xAVertex[j]=m_xAVertex[j]+N[i1][j]*Rx[i1];
			m_yAVertex[j]=m_yAVertex[j]+N[i1][j]*Ry[i1];
		}
	}
	for(i1=0;i1<=m_nCtrl0-2;i1++)
	{
		b[i1*2]=m_xAVertex[i1];
		b[i1*2+1]=m_yAVertex[i1];
	}
	if(agjdn(a,b,m_nCtrl0-1,2)!=0)
	{
		for(i1=1;i1<=m_nCtrl0-1;i1++)
		{
			m_xAVertex[i1]=b[(i1-1)*2];
			m_yAVertex[i1]=b[(i1-1)*2+1];
		}
```

```
                m_xAVertex[0]=m_xDVertex[0];
                m_yAVertex[0]=m_yDVertex[0];
                m_xAVertex[m_nCtrl0]=m_xDVertex[n];
                m_yAVertex[m_nCtrl0]=m_yDVertex[n];
        }
        else
        {
                Singular=true;
                return;
        }
}
```